VSR://EDU/SSE

Software Service Engineering

WS 2019/2020 – 3. Tutorial

Valentin Siegert M.Sc.

Shovra Das M.Sc.

VSR.Informatik.TU-Chemnitz.*de*

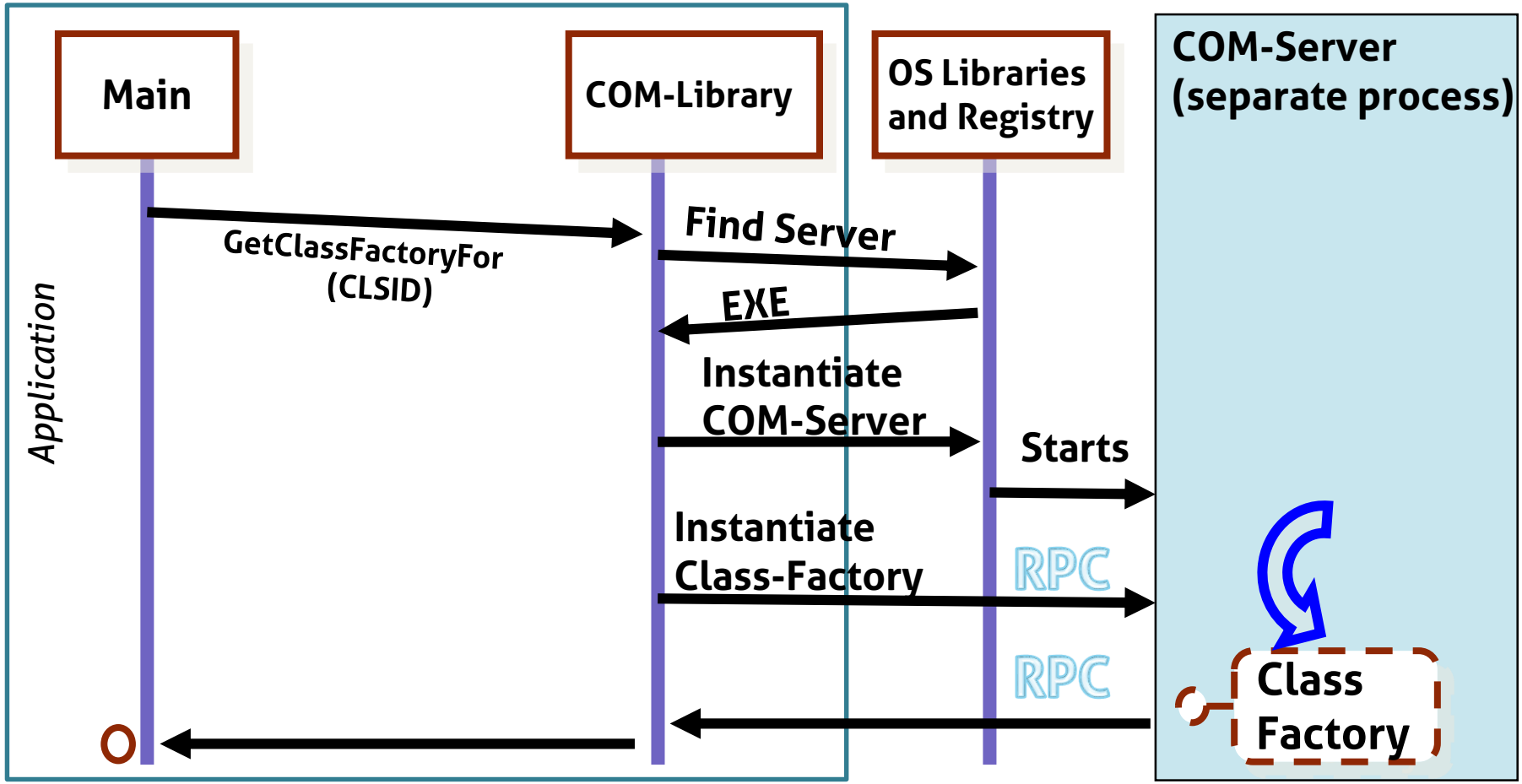
Homework Tutorial 2

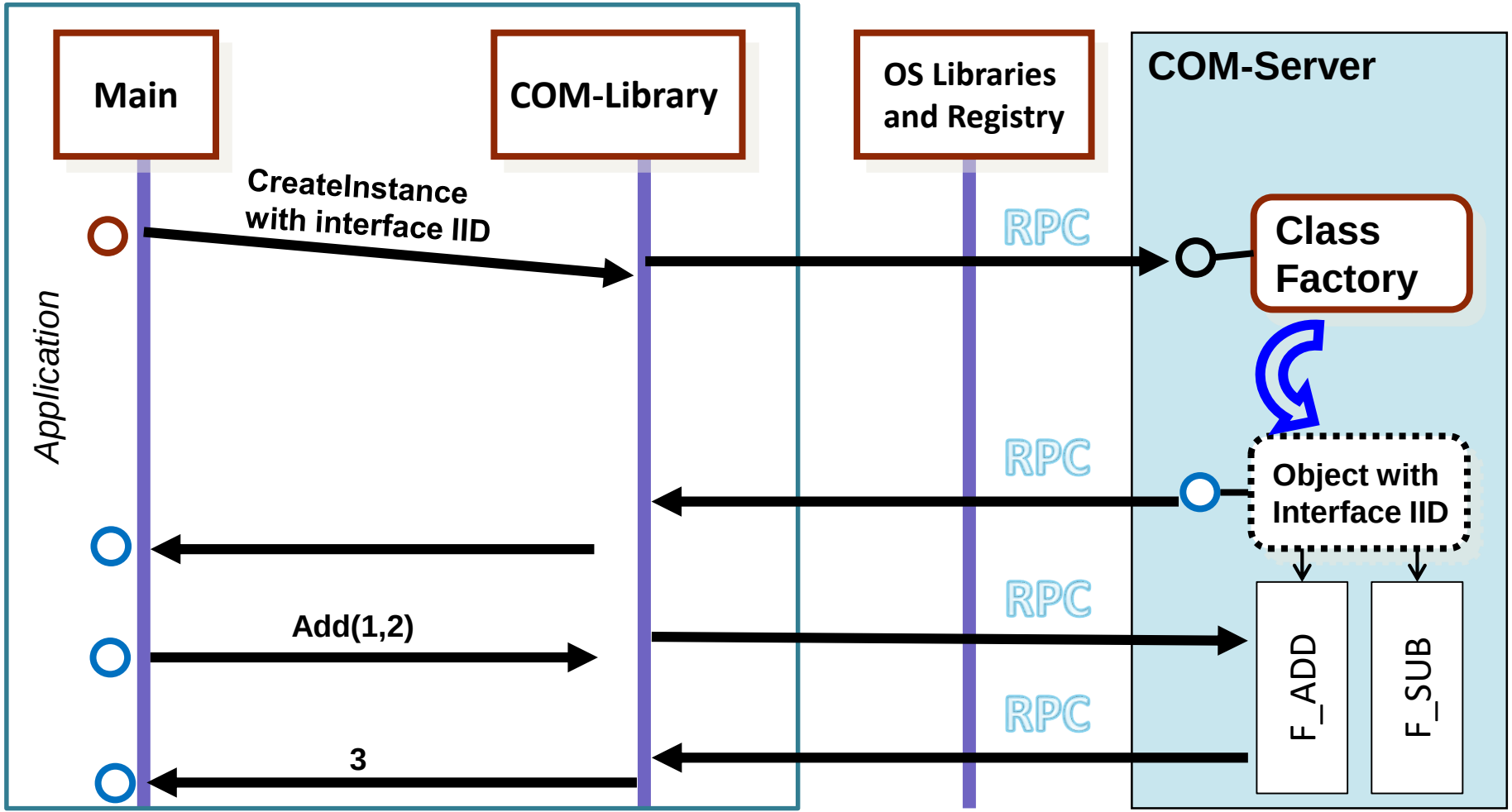
a) Describe the architecture of the Component Object Model (COM). What are the tasks of COM Component, COM Server, COM Registry and COM Interface?

b) Extract the project COM.zip. The subproject ComBrowserServer implements a COM component, which makes HTTP-GET requests to a given URL. ComBrowserClient is a C++ application, which makes use of the COM component.

- Microsoft's early standard for component technology
- Goal:
  - Application, platform, language and system independent code provisioning
- Involved elements:
  - COM Client
  - COM Server
    - In-Process
    - Out-Of-Process
  - COM Interface
  - COM Class
  - COM Object
  - COM Registry

- COM Interface
  - Declaration of functionalities
- COM Class
  - Binary code of some functionality (template for instantiation)
- Class ID (CLSID)
  - Globally Unique Identifier (GUID)
- COM Object
  - Instance of some COM class
- COM Server
  - Binary code containing COM classes (library or process)
- COM Registry:
  - Stores available COM servers
- COM Library:
  - COM Middleware


Main
COM-Library
OS Libraries and Registry
COM-Server (separate process)
Application
GetClassFactoryFor (CLSID)
Find Server
EXE
Instantiate COM-Server
Starts
Instantiate Class-Factory
RPC
RPC
Class Factory


Main
COM-Library
OS Libraries and Registry
COM-Server
Application
CreateInstance with interface IID
RPC
Class Factory
RPC
Object with Interface IID
Add(1,2)
RPC
F_ADD
F_SUB
RPC
3

Task 1

1. Answer the following questions:
   - What is the difference between URI and URL?

- URI is an abstract resource identifier (may be a unique name of the resource – URN or it‘s location – URL)
- URL describes a location of the resource and the protocol used to access it

1. Answer the following questions:
   - What is the difference between URI and URL?
   - What is the meaning of the URL scheme?

- URL scheme describes a method to access a resource
- Often (but not always) corresponds to some specific protocol:
  - http
  - ftp
  - news
  - ssh
  - file
  - ldap
  - …

1. Answer the following questions:
   - What is the difference between URI and URL?
   - What is the meaning of the URL scheme?
   - What, why and how should be encoded in URLS?

- What?
  - Segments of URLs
- Why?
  - Reserved characters : / ? # [ ] @ ! $ & % ' ( ) * + , ; =
  - Non-ASCII characters
  - Unsafe characters (whitespace, “,<,>...)
- How?
  - % + 2 Hexadecimal digits
  - Hexadecimal digits correspond to the ASCII-value of the character
  - Each byte of the UTF-8 encoding for non-ASCII symbols

2. Implement a class for parsing and generating URLs. Use the given template Task1-Template.zip as a start point:

- The constructor Url(string urlStr) should split the urlStr using a **regular expression** and fill in the instance variables Scheme, Host etc.
- The function string ToString() should concat the instance variables to the string representation of the URL.
- The static method string Encode(string s) should convert all characters from s, which are not in VALID_CHARACTERS into the %-form and give the resulting string back
- The static method string Decode(string s) should convert all %-escaped characters from sand give the resulting string back

Task 2

1. Explain the semantics of the following HTTP methods: GET, HEAD, PUT, DELETE, and POST. Which of them are **safe**, which are **idempotent** and which are **cacheable**?

Method Semantics

- GET – retrieve a resource
- HEAD – retrieve only resource metadata
- PUT – replace the resource with given representation
- DELETE – delete a resource
- POST – other actions

Method Characteristics

- A method is **safe** if it produces no side effects (no data is changed on the server-side)
- A method is **idempotent** if its multiple application yields the same side effects as if it was applied once (e.g. removal of a resource)
- A method is **cacheable** if the returned resources can be cached

	safe	idempotent	cachable
GET	✓	✓	(✓)
HEAD	✓	✓	(✓)
PUT	✗	✓	✗
DELETE	✗	✓	✗
POST	✗	✗	✗

1. Explain the semantics of the following HTTP methods: GET, HEAD, PUT, DELETE, and POST. Which of them are **safe**, which are **idempotent** and which are **cacheable**?
2. Explain the purpose of the following HTTP headers:
   a) Host
   b) Content-Type
   c) Content-Length
   d) Accept
   e) User-Agent
   f) Location

HTTP Headers

- **Host** - specifies virtual host and port number
- **Content-Type** – media-type of the resource representation
- **Content-Length** – length of the message body in bytes
- **Accept** – media-types supported by a client
- **User-Agent** – information about user's browser (agent in general)
- **Location** – information about new location of a resource

Task 3

Implement an HTTP message parser and builder based on the template Task3-Template.zip (take care of differentiation between request and response messages).
Complete the methods HTTPMessage(string) and ToString.

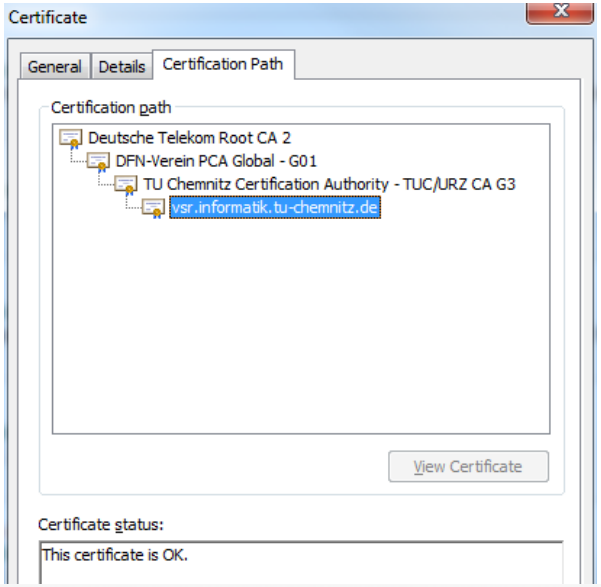
Task 4

1. What are the goals of HTTPS and how they are achieved?
2. What is the difference between HTTP and HTTPS request/response messages?

- In OSI-model in layer 6
- In TCP/IP-model
  - Above the Transport layer (i.e. TCP,...)
  - Below the Application layer (i.e. HTTP,...)
- Basic idea: generic security layer
- Protocol consists of 2 layers:

Handshake Protocol	Change Cipher Spec Protocol	Alert Protocol	Application Data Protocol
Record Protocol			

- Authentication using X509 certificates
- Authenticity of certificates is checked based on the Public Key Infrastructure (PKI)
- Encryption using asymmetric and symmetric algorithms
- Integrity using encrypted checksums



5 Homework

1. Inform yourself about the "chunked" transfer encoding and its purpose. Extend the HTTP message parser and builder from Task 3 with the support for “chunked” transfer encoding.
2. Based on the template Homework-Template.zip implement a server, which is able to deliver requested resources from the HttpServer.DOCUMENT_ROOT folder (for POST requests return only 201 Created). Test your implementation in the browser.
3. Modify the HTTP request implementation of Task 3 to request the following resource: https://www.tu-chemnitz.de (HTTPS)

VSR

Your feedback on today's session:

mytuc.org/tgxs

Questions?

valentin.siegert@informatik.tu-chemnitz.de

VSR.*Informatik*.TU-Chemnitz.*de*